

TITLE: METHOD AND SYSTEM FOR VIDEO SEQUENCE
REAL-TIME MOTION COMPENSATED TEMPORAL UPSAMPLING

5

CROSS-REFERENCE TO RELATED APPLICATION

This application claims benefit of United States Provisional Application Serial No. 60/405,414 entitled Method and System for Video Sequence Real-Time Motion Compensated Temporal Upsampling, filed August 22, 2002.

10

FIELD OF INVENTION

The invention relates to an improved method and system for video processing. In particular, this invention describes a method and system for motion-compensated temporal interpolation of video frames in processing video sequences such as computer video images, video games and other video effects where changes of an object's form or movement exist.

15

BACKGROUND OF THE INVENTION

This invention relates to encoding and decoding of complex video image information including motion components that may be found in multi-media applications such as video-conferencing, video-phone, and video games. In order to be able to transfer complex video information from one machine to another, it is often desirable or even necessary to employ video compression techniques. One significant approach to achieving a high compression ratio is to remove the temporal and spatial redundancy which is present in a video sequence. To remove spatial redundancy, an image can be divided into disjoint blocks of equal size. These blocks are then subjected to a transformation (e.g., Discrete

Cosine Transformation or DCT), which de-correlates the data so that it is represented as discrete frequency components.

Motion film and video provide a sequence of still pictures that creates the visual illusion of moving images. Providing that the pictures are acquired and displayed in an appropriate manner, the illusion can be very convincing. In modern television systems it is often necessary to process picture sequences from film or television cameras. Processing that changes the picture rate reveals the illusory nature of television. A typical example is the conversion between European and American television standards which have picture rates of 50 and 60 Hz respectively. Conversion between these standards requires the interpolation of new pictures intermediate in time between the input pictures. Many texts on signal processing described the interpolation of intermediate samples, for a properly sampled signal using linear filtering. Unfortunately, linear filtering techniques applied to television standards conversion may fail to work. Fast moving images can result in blurring or multiple images when television standards are converted using linear filtering because video signals are under-sampled.

The benefits of motion compensation as a way of overcoming the problems of processing moving images are widely recognized in the prior art. Motion compensation attempts to process moving images in the same way as the human visual system. The human visual system is able to move the eyes to track moving objects, thereby keeping their image stationary on the retina. Motion compensation in video image processing attempts to work in the same way. Corresponding points on moving objects are treated as stationary thus avoiding the problems of under sampling. In order to do this, an assumption is made that the image consists of linearly moving rigid objects (sometimes slightly less restrictive

assumptions can be made). In order to apply motion-compensated processing it is necessary to track the motion of the moving objects in an image. Many techniques are available to estimate the motion present in image sequences.

Motion compensation has been demonstrated to provide improvement in the

5 quality of processed pictures. The artifacts of standard conversion using linear filtering, i.e., blurring and multiple-imaging, can be completely eliminated. Motion compensation, however, can only work when the underlying assumptions of a given subject are valid. If a subject image does not consist of linearly moving rigid objects, the motion estimation and compensation system is unable to reliably track motion resulting in random motion vectors.

10 When a motion estimation system fails, the processed pictures can contain subjectively objectionable switching artifacts. Such artifacts can be significantly worse than the linear standards conversion artifacts which motion compensation is intended to avoid.

Motion vectors are used in a broad range of video signal applications, such as coding, noise reduction, and scan rate or frame-rate conversion. Some of these applications,

15 particularly frame rate conversion, requires estimation of the “true-motion” of the objects within a video sequence. Several algorithms have been previously proposed to achieve true-motion estimation. Algorithms have also been proposed that seek to provide motion estimation at a low complexity level. Pel-recursive algorithms that generally provide sub-pixel accuracy, and a number of block-matching algorithms have been reported that yield

20 highly accurate motion vectors.

SUMMARY OF THE INVENTION

The present invention provides a video signal processing method and system for

reconstruction of a previously compressed video sequence by creation and insertion of supplementary video frames into an upsampled video sequence. Movement characteristics of an object within a video frame are evaluated and used to estimate the object's position in the supplementary video frames.

5 Motion vectors are determined for object motion within video frames using the object's position information from the two neighboring frames, which are then used for the filling in new frames. After the creation of the new frame, a border is re-established along the edge of the frame.

10 The present invention additionally provides a means of removal of the dark bands present along the edges of video frames by extrapolating a video frame – inserting blocks of video information beyond the borders of the real frame. This broadening of frames beyond the original frame borders also provides for better interpolation of new frames.

BRIEF DESCRIPTION OF THE DRAWINGS

15 Figure 1 illustrates the insertion of additional frames into the video sequence.

Figure 2 is a block diagram representation of the operation of the temporal interpolation algorithm on a video sequence.

Figure 3 illustrates a example of the movement of an object in a video film.

Figure 4(a) shows an example of a dark band along the edge of a video frame.

20 Figure 4(b) illustrates the filling in of the dark band along the edge of a video frame with pixels from within the frame.

Figure 5 illustrates an extrapolation of a video frame beyond its original borders.

Figure 6 illustrates the application of the present invention on an example of frames from a “Foreman” video sequence.

Figure 7 illustrates the application of the present invention on an example of frames from a “Football” video sequence.

5 Figure 8 illustrates details of the broadening of blocks in a frame.

Figure 9 shows the order of passing of blocks within a video frame.

Figure 10 shows the structure of a block search within a video frame.

DETAILED DESCRIPTION OF THE INVENTION

10 The following definitions apply to the present invention:

Temporal interpolation: a process of generation and insertion of one or more video frames into a source video sequence.

Interpolated frame: a video frame generated by the present invention method that is inserted between the existing frames of a video sequence.

15 Object motion: the change in location of an object over the course of a video sequence.

Object deformation: the change in form of an object over the course of a video sequence.

20 Motion vector: the difference between the coordinates of the block in the preceding frame and the coordinates of the corresponding block found in the next frame.

Motion estimation: the process of locating motion vectors for all blocks in the preceding frame.

Adjacent frames: sequential video frames with the numbers N and N+1 in a video sequence, as shown in Figure 1.

Fps: the number of frames per second for a video sequence. The present invention permits a variation of the value of fps by a whole number for the source video
5 sequence.

Extrapolation: the broadening of a video frame beyond its original borders by a specified number of pixels.

Application of the present invention algorithm creates an arbitrary number of frames between two adjacent frames of a video sequence. Depending on the dynamic of the
10 subject of a video sequence, the need may arise for the insertion of one frame or more video frames between two adjacent frames. The insertion of frames helps to produce smoothness of the movements of objects in the video film. There are different types of dynamics for development of the subject of a video sequence. In drawings 6 and 7, frames from the video sequences “Foreman” and “Football” are provided as examples of video sequences with a
15 high dynamic of subject development; the objects exhibit a high degree of movement for several frames, and a significant change of scene or subject takes place.

Figure 6 shows the addition of several video frames between adjacent frames in the “Foreman” video sequence. Interpolated frames E and F were inserted between original frames D and G in order to smooth the transition from one scene to another and to make the
20 movements of the subject more realistic.

Video frames from the “Football” video sequence shown in Figure 7 presents a somewhat different situation. There is also rapid movement of objects here, but the background scene does not change. A video frame is added in this case in order to make the

movement of the objects (the falling of football players with a ball) more natural.

The algorithm provides a means for:

1. Reading two adjacent video frames in a sequence.
2. Identifying and storing the dark band along the edge of the frame (if present), and removing the band from the frame as illustrated in Figure 5
3. Expanding each video frame beyond its original borders by 8 pixels from each side of the frame, as shown in Figure 5, applying the following extrapolation procedure.
4. Splitting the next video frame into quadratic blocks, and finding the corresponding block within the previous frame in order to obtain a displacement vector for each block.
5. Decreasing the length of the displacement vector by k/n times, where n is the coefficient for increasing fps, and $0 < k < n$ is a number relative to the previously inserted frame.
6. Determining the presence of a scene change.
7. If there was no scene change, then the interpolated frame is filled using the calculated movement vectors.
8. If there was a scene change, then the pixels of the interpolating frame are filled using values for the type of interpolation applied.
9. Restoring the dark band along the edges of each video frame if a dark band was originally present.
10. Creating the interpolated video frame and inserting it into the up-sampled

video sequence as the video output.

In some video sequences there is a dark band along the edges of the video frames as illustrated in figures 4(a) and 4(b). The following is a description of the procedure for detection and removal of defects (dark bands) along the edges of a video frame.

5 The dark band consist of bands which we perceive as black and bands, the brightness of which differs significantly from the brightness of the primary part of the frame. The average brightness value of the black band in the YUV color system does not exceed 20, and the brightness differential for the darkened band is not less than 35. The difference between the average brightness of the dark band and the average brightness of the
10 subsequent band of pixels, as a rule, is more than 35. It is necessary to remove these bands in order to allow extrapolation of a video frame beyond its original borders.

The algorithm for detection and removal of video frame defects (dark bands) along the edges of the frame provides the following steps:

1. Calculating the average brightness values for m bands of a width of 1 pixel.

15
$$\text{AverageY}(x) = \left(\sum_{y=0}^{\text{height}-1} Y(x, y) \right) / \text{height}$$

- for vertical bands

$$\text{AverageY}(y) = \left(\sum_{x=0}^{\text{width}-1} Y(x, y) \right) / \text{width}$$

- for horizontal bands.

2. A band is considered dark if the following conditions are fulfilled:

if

20 $\text{AverageY}[i] < 20$

or

$$(\text{AverageY}[i+1] - \text{AverageY}[i]) > 35.$$

3. The brightness values for pixels of a dark band are replaced by values for
5 the pixels from the first non-dark band encountered, as shown in Figure 4(b).

An extrapolation algorithm is applied for predicting the brightness of pixels outside the borders of a frame of a video image following the procedure for removal of the dark band (if present.) An extrapolation filter of length 4 is used.

Input data:

10	Image	Input image (frame)
	Height	Frame height
	Width	Frame width
	Number_of_points	Number of interpolation points
	I_1, I_2, I_3, I_4	Reference points
15	k_1, k_2, k_3, k_4	Filter coefficients
	I_1	Extrapolated point

Output data:

20	Extrapolated_Image	Extrapolated image
----	--------------------	--------------------

The extrapolation algorithm performs the following steps:

1. The image is transformed into YUV video format, and the algorithm is applied to each layer.
2. Four reference points are established for the filter. New reference points are then determined in relationship to these four points.
3. If the base points I_1 , I_2 , I_3 , I_4 are established, then I_0 is a point that will be extrapolated.
4. Let k_1 , k_2 , k_3 , k_4 be the filter coefficients. The corresponding extrapolated point is then calculated by the following method:

$$I_0 = \frac{\sum_{i=1}^4 I_i * k_i}{\sum_{i=1}^4 k_i}$$

10

5. Selection of an appropriate filter coefficient for use in the algorithm is critical. The largest coefficient k_1 is selected, and is increased to the brightness value of the outermost pixel of the frame, as shown in Figure 5.

Broadening blocks within a video frame requires selection of the most appropriate (“best”) block for the given video frame. It is important to apply a metric that allows the detection a compatibility of blocks and that does not require the use of vast computer resources. The following metric is used in the present invention:

$$\begin{aligned}
SAD = & SAD(Y_{x_0, y_0}, \hat{Y}_{x_1, y_1}, \text{block_size_x}, \text{block_size_y}) + \\
& 4 \cdot SAD(U_{x_0/2, y_0/2}, \hat{U}_{x_1/2, y_1/2}, \text{block_size_x}/2, \text{block_size_y}/2) + \\
& 4 \cdot SAD(V_{x_0/2, y_0/2}, \hat{V}_{x_1/2, y_1/2}, \text{block_size_x}/2, \text{block_size_y}/2)
\end{aligned}$$

where

$$\begin{aligned}
SAD(I_{x_0, y_0}, \hat{I}_{x_1, y_1}, \text{block_size_x}, \text{block_size_y}) = \\
\sum_{i=0}^{\text{block_size_y}} \sum_{j=0}^{\text{block_size_x}} \left| I_{x_0+i, y_0+j} - \hat{I}_{x_1+i, y_1+j} \right|
\end{aligned}$$

where I_{x_0, y_0} and \hat{I}_{x_1, y_1} are comparable blocks from frames I and \hat{I} .

5 The coordinates for the blocks are (x_0, y_0) and (x_1, y_1) , respectively. The given coordinates are coordinates for the left upper corner of the block.

`block_size_x` and `block_size_y` – are the measurements of the blocks.

`Y` - luminance, `U`, `V` -chrominance.

The following describes the procedure for splitting the frame into quadratic blocks.

10 Prior to execution of the search, the next frame is covered with non-overlapping quadratic blocks of size $N \times N$, where N is selected depending on the size of the frame:

Frame format	Frame size in pixels (Height)	N
QCIF	144	4
SIF	240	8
BT	480	16
R601	480	16

During a search of the vectors a comparison is conducted for 16x16 blocks obtained from NxN blocks by means of adding to them a layer of surrounding pixels, as shown in Figure 8.

Depending on the frame sizes the following conditions are developed:

10 N=4;
 if (height>200)
 {N=8} if (height>400); and
 {N=16}

15 Both frame height and width are verified. Thus, depending on the format of the source video sequence, system parameters are automatically adjusted.

Calculation of the coordinates of the lower left corner of a 16x16 block is provided as follows:

Input data:

x, y	Coordinates for the lower left corner of a NxN block
x_1, y_2	Coordinates for the lower left corner of a 16x16 block
width	Frame width
height	Frame height

The algorithm for calculation of the coordinates of the lower left corner of a 16x16 block:

$$1. \quad x_1 = x - (16-N)/2;$$

$$2. \quad y_1 = y - (16-N)/2;$$

10 Verification is made of the following conditions:

$$\text{if } x_1 < 0 \Rightarrow x_1 = x;$$

$$\text{if } y_1 < 0 \Rightarrow y_1 = y;$$

$$\text{if } x_1 + (16-N) > \text{width} - 1 \Rightarrow x_1 = x - (16-N);$$

$$\text{if } y_1 + (16-N) > \text{height} - 1 \Rightarrow y_1 = y - (16-N);$$

15 where x, y are coordinates for block NxN;

where x_1, y_2 are coordinates for block 16x16;

width – frame width;

height – frame height.

The following sequence for searching blocks for displacement vectors has been found

20 to be optimal.

1. Beginning with a central block, an outward-spiraling by-pass of the blocks is made, as illustrated in Figure 9.

2. The displacement vector for the remaining blocks is then calculated.

An adaptive accelerated search for the displacement vectors of blocks is applied separately for each block:

The algorithm for the search procedure is intended to search zones, use a vector-predictor from adjacent blocks and the preceding frame, establishing criteria for a half-pause, and provide adaptive selection of thresholds.

An aggregate of thresholds (T1, T2, T3) is used to control the course of the search operation as follows:

Threshold	Threshold designation
T1	Determines the continuation or completion of the search
T2	Determines if the number of subsequent examined/verified search zones is limited
T3	Determines the completion of the search according to the half-pause criterion

The following variables are also applied to the search operation:

zsize	The parameter for the half-pause criterion, gives the maximum number of search zones during the scanning of which a mistake in a located vector may not be corrected;
znum	The maximum number of search zones around the selected center (0, 0);
pznum	The maximum number of search zones around the main vector-predictor (median);
MinZone	The current number of zones in which a vector was found with minimal error;
Found	An indication of the fact that all vector-predictors are equal to each other, different from (0,0), and correspond to the vector of the block positioned in the previous frame in the current position;
Last	An indicator of the final iteration;
MinSAD	The current located minimal error;

10 The following are the initial values for the thresholds T1, T2 and T3 and the associated variables described above:

T1	T2	T3	zsize	znum	pznum	MinZone	Found	Last
4*256	6*256	14*256	3	4	4	0	false	false

15

The following are the algorithm steps for the search procedure:

Step 1

Building a rhomboid-shaped structure containing a selected number of search zones, as shown in Figure 10.

20

Step 2

Selecting and storing adjacent blocks for the processed block, in which displacement vectors have already been found are selected and stored. The selected blocks are sorted according to the increment of errors (SAD). Blocks in which there is an error twice as large as the smallest error are eliminated. An aggregate of predictors for the search is thus created

5 – aggregate A containing the vectors for these blocks.

Step 3

Calculating the threshold values. Threshold T1 is selected as the minimum from the error values (SAD) for the adjacent blocks, selected in step 2, and the error values for the
10 block from the preceding frame in the current position of the splitting array.

$T_2 = T_1 + \text{the dimension of the block in pixels}$. The algorithm parameters are thus initialized.

Step 4

15 Calculating the median of the vectors for the x and y coordinates for the selected adjacent blocks. If the values of all vectors in the aggregate of predictors A:

- 1) coincide and are different from $(0,i)$ and $(i,0)$, where i is a whole number; and
- 2) coincide with the value of the median,

20 then the search will be conducted only in the first zone ($pznum = 1$) and the “Found” character is specified. If only one of these conditions is fulfilled, then the search will be conducted only in the first two zones ($pznum = 2$). The predictor forecasts the character of the movement in the given place in the interpolated frame. Due to the determination of the

predictor, the system is adjusted to accommodate the determined character of movement in the given adjacent frames of the video sequence, used for the interpolation of new frames.

Step 5

5 Calculating the error (SAD) for the main predictor. If the main predictor coincides with the vector of the block positioned in the preceding frame in the same position as the main predictor, but in this case the predictor error (SAD) is less than the error for the indicated vector, or the error according to the value is less than the dimensions of the block, then skipping to the final step.

10

Step 6

Calculating the error (SAD) for all vectors in the aggregate A, and selecting with the current value the vector with the minimal error MinSAD.

15 Step 7

Verifying the condition $\text{MinSAD} < T_1$. If the condition is satisfied, the process skips to the final step. If the current vector coincides with the vector for the block located in the previous frame in the same position, but the current minimal error in this case is less, the process also skips to the final step.

20

Step 8

If $T_1 < \text{Min SAD} < T_2$, then we in fact establish the character “Last”.

Step 9

Constructing a given number of zones around the main predictor. Then, in the subsequent steps, processing in sequence each of the constructed zones beginning from the center.

5

Step 10

Calculating the error (SAD) for all points from each of the zones.

Step 11

10 Verifying that the result was improved within the framework of the given number of the nearest already-processed zones zsize. If the improvements were made previously (the current zone – MinZone > size), and MinSAD < T3, then the process skips to the final step.

Step 12

15 If there is no minimal error value in the second search zone, and MinSAD < T3, then the process skips to the final step.

Step 13

20 If MinSAD < T1 or in fact is the character “Last”, then the process skips to the final step.

Step 14

Returning to step 8 as often as required.

Step 15

Shifting to processing the next furthest from the center zone, and processing to step 10.

5 Step 16

Steps 9 to 15 are repeated, but this time the center zone moves to the point (0,0).

Step 17

Steps 9 to 14 are repeated, but this time the center zone moves to the point, the 10 coordinates of which are given by the best vector found up to that moment.

Step 18

Obtaining the optimal vector of movement for the given block, having a minimal error MinSAD.

15

Description of the procedure for calculating the values of pixels with non-integer coordinates.

The given procedure is carried out in the event that the block has a displacement vector with non-integer coordinates, that is it is displaced by a fractional number of pixels.

20 Thus, the necessity for calculation of the values of the intermediate pixels for blocks located in the original adjacent frames of the video sequence arises. The values of the intermediate pixels are calculated with the help of a bilinear interpolation formula:

$$I(x+dx, y+dy) = I(x, y) \cdot (1-dx) \cdot (1-dy) + I(x+1, y) \cdot dx \cdot (1-dy) + I(x, y+1) \cdot (1-dx) \cdot dy + I(x+1, y+1) \cdot dx \cdot dy$$

$I(x+1, y+1) \cdot dx \cdot dy;$

Where $I(x, y)$ - is the value of the pixel,

x, y – are the coordinates of the pixel.

The obtained result is rounded up to the nearest whole number.

5 The following are the algorithm steps for filling an interpolated frame on the basis of
the calculated vectors of movement.

1. Superimposing on an interpolating frame the same array NxN as in the subsequent frame.
2. For each block, applying the displacement vector obtained from the adaptive accelerated search procedure.

10 Filling in the image pixels with the following values :

$$I_{\text{interp}}(x, y, vx, vy, k, n) = ((n-k) \cdot I_{\text{prev}}(x+k \cdot vx/n, y+k \cdot vy/n) + k \cdot I_{\text{next}}(x-(n-k) \cdot vx/n, y-(n-k) \cdot vy/n))/n, \text{ where}$$

n is the coefficient for increasing fps;

15 0 < k < n is the number relative to the preceding, inserted frame;

I_{interp} are the points of the interpolated frame;

I_{prev} – are the points of the preceding frame;

I_{next} – are the points of the subsequent frame;

x, y – are the coordinates for the pixel in the interpolated frame;

20 vx, vy – are the coordinates for the displacement vector, as determined by the adaptive accelerated search procedure.

Applying a bilinear interpolation for points with a fractional vector of displacement.

A video sequence can be seen as an aggregate of scenes. A scene is a part of a video

sequence in which it is possible to form each subsequent frame on the basis of the previous frame with the assistance of a motion estimation procedure. If the presence of a change of scene is detected, then either the preceding or subsequent frame is duplicated.

The following algorithm variables are used for determining the presence of a scene
 5 change:

cnt_bad_blk	The number of vectors with an error less than some threshold
block_cnt	The number of blocks into which the frame is split
err[i]	The error for the I block
scale	The coefficient for increasing fps
10 pos	The number for the inserted frame relative to the preceding frame, with which the reference is begun with zero.

The following is the algorithm steps are used for determining the presence of a scene
 change:

```

int cnt_bad_blk=0;
15
for (int i=0; i< block_cnt;i++)
{
    if(err[i]>15*256[i])
    {
        cnt_bad_blk++;
    }
}
20
25
bool scene_is_changed = (cnt_bad_blk>0.7*block_cnt);
```

If the variable `scene_is_changed` has a positive value, then a change of scene is considered to have taken place. Either the preceding or the subsequent frame is then inserted as the interpolating frame, and is followed by the following:

```
if(scene_is_changed)
5
{
    if(pos<scale/2)
    {
        the preceding frame is inserted
    }
10
else
{
    the subsequent frame is inserted
}
}
15
```

A decision must be made as to the type of interpolation for a given segment of the video sequence. An interpolation type is implemented using motion vectors (motion compensated interpolation), or a pixel by pixel interpolation.

Pixel-by-pixel interpolation is carried out using the following formula:

$$20 \quad I_{\text{interp}}(x, y, k, n) = ((n-k) \cdot I_{\text{prev}}(x, y) + k \cdot I_{\text{next}}(x, y)) / n,$$

where `I_interp` is the value of the interpolating pixel;

`I_prev` – is the value of the pixel in the preceding frame;

`I_next` – is the value of the pixel in the next frame;

`x, y` - are the pixel coordinates;

25 `n` – is the coefficient for increasing fps;

`k` – is the number relating to the preceding, inserted frame, $0 < k < n$.

The following variables are used in the algorithm for determining the presence of a scene change:

	cnt	The number of vectors with an error less than some threshold
5	average_v_x, average_v_y	Coordinates of the average vector
	block_cnt	The number of blocks into which a frame is split
	vx[i], vy[i]	Coordinates of the displacement vector for block i.
	err[i]	The error for block i
10	max_v_x	The maximum value for the x -components of a vector with an error greater than some threshold
	max_v_y	The maximum value for the y-components of a vector with an error greater than some threshold
	maxv	The larger of the two values max_v_x and max_v_y

The average vector is calculated using the following algorithm:

```

15      cnt=0
          average_v_x=0
          average_v_y=0

20      for (int i=0; i<block_cnt; i++)
          {
              if(err[i]<15* 256[i])
              {
                  average_v_x+=vx[i];
                  average_v_y+=vy[i];
              }
          }
      
```

```

        cnt++;
    }
}

average_v.x/=cnt;
5
average_v.y/=cnt;

```

The following algorithm is used to determine the type of interpolation to be applied:

```

max_v_y=0;
10
for (int i=0; i<block_cnt;i++)
{
    if(err[i]<15*256[i])
    {
        cnt++;
    }
    if(err[i]>15* 256[i])
    {
        if(abs(vy[i])>max_v_y)
20
        {
            max_v_y=abs(vy [i]);
        }
    }
    if(abs(vx [i])>max_v_x)
25
    {
        max_v_x=abs(vx [i]);
    }
}

```

```
    }  
    max_v_x=abs(max_v_x-abs(average_v_x));  
    max_v_y=abs(max_v.y-abs(average_v_y));  
    maxv=max(max_v_x, max_v_y)  
5     not_use_MCI =  
(maxv>(Segm_img>height/32)&&abs(average_v_x)<1&&abs(average_v_y)<1)
```

If the value of the variable not_use_MCI is positive, then the pixel-by-pixel interpolation described above is applied. Otherwise, interpolation using vectors of movement (motion compensated interpolation) will be applied.

INDUSTRIAL APPLICABILITY

The present invention has applicability to the field of video processing, and more particularly to a method and system for temporal interpolation of video sequences in an electronic processing, transmission or storage medium.

In compliance with the statute, the invention has been described in language more or less specific as to algorithm steps in processing video sequences. It is to be understood, however, that the invention is not limited to the specific means, features or algorithms shown or described, since the means, features and algorithms shown or described comprise preferred ways of putting the invention into effect.

Additionally, while this invention is described in terms of being used for temporal interpolation of video sequences, it will be readily apparent to those skilled in the art that the invention can be adapted to other uses for other forms of digital electronic signal processing as well, and therefore the invention should not be construed as being limited to processing video sequences. The invention is, therefore, claimed in any of its forms or modifications

within the legitimate and valid scope of the appended claims, appropriately interpreted in accordance with the doctrine of equivalents.